Reinforcement Learning

Policy Gradient Methods

Stefano V. Albrecht
28 February 2023
Lecture Outline

- Parameterised policies
- Softmax and Gaussian policies as examples
- Policy gradient theorem
- Policy gradient methods: REINFORCE, baselines, actor-critic family
Previously: approximate value function with parameterised function

\[
\hat{v}(s, w) \approx v_\pi(s) \\
\hat{q}(s, a, w) \approx q_\pi(s, a)
\]

Policy was generated *implicitly* from value function (e.g. $\epsilon$-greedy)

- **Compact**: number of parameters in $w$ can be much smaller than $|S|$
- **Generalises**: changing one parameter value may change value of many states/actions
Today: approximate policy with parameterised function

$$\pi(a|s, \theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$$

$\theta \in \mathbb{R}^{d''}$ is policy parameter vector

e.g. linear function, neural network, decision tree, ...

- **Compact**: number of parameters in $\theta$ can be much smaller than $|S|$
- **Generalises**: changing one parameter value may change action in many states
Advantages of optimising policy directly:

- Can learn **stochastic policies** (assign any probabilities to actions)
- Effective in high-dimensional and **continuous action spaces**  
  ⇒ Important for robotics applications
- Better convergence properties, typically to local optimum
Example: Optimal Stochastic Policy in Short Corridor

Reward is $-1$ until goal state reached

Assume agent cannot distinguish between states, only between left/right action

$\epsilon = 0.1$
How to parameterise policy?

- We focus on gradient-based optimisation → need differentiable policy
- Examples:
  - Softmax for discrete actions
  - Gaussian for continuous actions
  - Deep neural network (next lectures)
How to parameterise policy?

- We focus on gradient-based optimisation → need differentiable policy
- Examples:
  - Softmax for discrete actions
  - Gaussian for continuous actions
  - Deep neural network (next lectures)

How to optimise policy parameters?

- Policy gradient theorem leads to family of optimisation algorithms
- Monte Carlo, n-step TD, TD(λ), ...
For discrete actions, can use **softmax** policy:

$$\pi(a|s, \theta) \doteq \frac{e^{h(s, a, \theta)}}{\sum_b e^{h(s, b, \theta)}}$$

- Action preference $h(s, a, \theta)$ can be parameterised arbitrarily, e.g. linear in features

$$h(s, a, \theta) = \theta^\top x(s, a)$$
Gaussian Policy for Continuous Actions

For continuous actions, can use Gaussian policy:

\[ a \sim \mathcal{N}(\mu(s, \theta), \sigma^2) \]

- Mean \( \mu \) can be parameterised arbitrarily, e.g. linear in features

\[ \mu(s, \theta) = \theta^T x(s) \]

- Variance \( \sigma^2 \) can be fixed or also parameterised (see book)
Goal: given policy representation $\pi(a|s, \theta)$, find optimal parameters $\theta$

How to measure quality of $\theta$?

- In episodic tasks, can use value of start state $s_0$:
  \[ J(\theta) \doteq v_{\pi_\theta}(s_0) \]

- In continuing tasks, can use average reward:
  \[ J(\theta) \doteq \sum_s P_\pi(s) \sum_a \pi(a|s, \theta) \sum_{s', r} p(s', r|s, a) r \]

$P_\pi(s)$ is steady-state distribution under $\pi$
Policy Gradient

- Policy gradient algorithms search for a *local* maximum in $J(\theta)$ by ascending the gradient of $\pi$ wrt $\theta$

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

- $\nabla J(\theta)$ is the *policy gradient*

$$\nabla J(\theta) = \left( \frac{\partial J(\theta)}{\partial \theta_1}, \ldots, \frac{\partial J(\theta)}{\partial \theta_{d'}} \right)$$
Policy Gradient Theorem:
For any differentiable policy $\pi$, the policy gradient is

$$\nabla J(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta)$$

$d_\pi(s)$ is the on-policy distribution under $\pi$:

- For start-state value: $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr\{S_t = s | s_0, \pi\}$
- For average reward: $d_\pi(s) = \lim_{t \to \infty} \Pr\{S_t = s | \pi\}$ (steady-state dist.)

**Note:** does not require derivative of environment dynamics $p(s', r|s, a)$!
Since $d_\pi(s)$ is on-policy, we can sample approximate gradient:

$$
\nabla J(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \theta)
$$

$$
= \mathbb{E}_\pi \left[ \sum_a q_\pi(S_t, a) \nabla \pi(a|S_t, \theta) \right]
$$

$$
= \mathbb{E}_\pi \left[ \sum_a \pi(a|S_t, \theta) q_\pi(S_t, a) \frac{\nabla \pi(a|S_t, \theta)}{\pi(a|S_t, \theta)} \right]
$$

$$
= \mathbb{E}_\pi \left[ q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \right]
$$

$$
= \mathbb{E}_\pi \left[ q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta) \right]
$$
General Gradient Update

**General gradient update:** \( \theta_{t+1} = \theta_t + \alpha (q_{\pi}(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta_t)) \)

A policy gradient method needs to:
General Gradient Update

General gradient update:  $\theta_{t+1} = \theta_t + \alpha(q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta_t))$

A policy gradient method needs to:

- Compute/approximate $\nabla \ln \pi(A_t|S_t, \theta_t)$
  - Softmax policy: $\nabla \ln \pi(a|s, \theta) = x(s, a) - \sum_{a'} \pi(a'|s, \theta) x(s, a')$
  - Gaussian policy: $\nabla \ln \pi(a|s, \theta) = (a - \mu(s, \theta)) x(s) / \sigma^2$

Approximate $q(S_t; A_t)$ e.g. Monte Carlo: use $G_t$, since $E[G_t|S_t; A_t] = q(S_t; A_t)$
General gradient update: \( \theta_{t+1} = \theta_t + \alpha \left( q_\pi(S_t, A_t) \nabla \ln \pi(A_t|S_t, \theta_t) \right) \)

A policy gradient method needs to:

- Compute/approximate \( \nabla \ln \pi(A_t|S_t, \theta_t) \)
  
  \(-\) Softmax policy: \( \nabla \ln \pi(a|s, \theta) = x(s, a) - \sum_{a'} \pi(a'|s, \theta) x(s, a') \)
  
  \(-\) Gaussian policy: \( \nabla \ln \pi(a|s, \theta) = (a - \mu(s, \theta)) x(s) / \sigma^2 \)

- Approximate \( q_\pi(S_t, A_t) \)
  
e.g. Monte Carlo: use \( G_t \), since \( \mathbb{E}_{\pi}[G_t|S_t, A_t] = q_\pi(S_t, A_t) \)

\( \Rightarrow \) REINFORCE algorithm
REINFORCE Update Rule

REINFORCE update rule:

\[ \theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \]

- \( \nabla \pi(A_t|S_t, \theta_t) \)
- \( G_t \)
- \( \pi(A_t|S_t, \theta_t)^{-1} \)
REINFORCE Update Rule

REINFORCE update rule:

\[ \theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \]

- \( \nabla \pi(A_t|S_t, \theta_t) \) – direction in parameter space that most increases the probability of repeating action \( A_t \) on future visits to state \( S_t \)

- \( G_t \)

- \( \pi(A_t|S_t, \theta_t)^{-1} \)
REINFORCE Update Rule

REINFORCE update rule:

\[ \theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \]

- \( \nabla \pi(A_t|S_t, \theta_t) \) – direction in parameter space that most increases the probability of repeating action \( A_t \) on future visits to state \( S_t \)
- \( G_t \) – make gradient magnitude proportional to return (better actions get larger updates)
- \( \pi(A_t|S_t, \theta_t)^{-1} \)
REINFORCE update rule:

\[ \theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta_t)}{\pi(A_t|S_t, \theta_t)} \]

- \( \nabla \pi(A_t|S_t, \theta_t) \) – direction in parameter space that most increases the probability of repeating action \( A_t \) on future visits to state \( S_t \)
- \( G_t \) – make gradient magnitude proportional to return (better actions get larger updates)
- \( \pi(A_t|S_t, \theta_t)^{-1} \) – make gradient magnitude inversely proportional to probability of \( A_t \) to normalise against frequency of observed \( A_t \) (like importance sampling)
REINFORCE Pseudocode

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ (e.g., to $0$)

Loop forever (for each episode):

Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \theta)$

Loop for each step of the episode $t = 0, 1, \ldots, T - 1$:

\[
G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k
\]
\[
\theta \leftarrow \theta + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \theta)
\]
$G_0$
Total reward on episode averaged over 100 runs

$\alpha = 2^{-13}$
$\alpha = 2^{-14}$
$\alpha = 2^{-12}$
Baseline to Reduce Variance in Updates

Can generalise policy update to include baseline:

\[(q_\pi(S_t, A_t) - b(S_t)) \nabla \ln \pi(A_t|S_t, \theta)\]

Does not change expectation:

\[
E_\pi[\nabla \ln \pi(A_t|S_t, \theta) b(S_t)] = \sum_s d_\pi(s) \sum_a \nabla \pi(a|s, \theta) b(s) \\
= \sum_s d_\pi(s) b(s) \nabla \sum_a \pi(a|s, \theta) \\
= \sum_s d_\pi(s) b(s) \nabla 1 = 0
\]

But can reduce variance of updates, e.g. use \(b(s) = \hat{v}(S_t, w)\)
REINFORCE with Baseline in Corridor Example

\[ \hat{v}(S_t, w) = w \] (one parameter in \( \hat{v} \))
Actor-Critic Methods

REINFORCE uses MC updates:

- large variance in updates (as any MC method)
- has to wait until end of episode (as any MC method)

Policy gradient can also use TD methods → then called **Actor-Critic method**

e.g. semi-gradient TD(0):

\[
\theta_{t+1} = \theta_t + \alpha (R_{t+1} + \gamma \hat{V}(S_{t+1}, w) - \hat{V}(S_t, w) ) \nabla \ln \pi(A_t|S_t, \theta)
\]

- **Critic** updates value function parameters \( w \)
- **Actor** updates policy function parameters \( \theta \)
Actor-Critic with Semi-Gradient TD(0)

Input: a differentiable policy parameterization $\pi(a|s, \theta)$
Input: a differentiable state-value function parameterization $\hat{v}(s,w)$
Parameters: step sizes $\alpha^\theta > 0$, $\alpha^w > 0$
Initialize policy parameter $\theta \in \mathbb{R}^{d'}$ and state-value weights $w \in \mathbb{R}^d$ (e.g., to 0)
Loop forever (for each episode):
  Initialize $S$ (first state of episode)
  $I \leftarrow 1$
  Loop while $S$ is not terminal (for each time step):
    $A \sim \pi(\cdot|S, \theta)$
    Take action $A$, observe $S', R$
    $\delta \leftarrow R + \gamma \hat{v}(S',w) - \hat{v}(S,w)$ (if $S'$ is terminal, then $\hat{v}(S',w) \equiv 0$)
    $w \leftarrow w + \alpha^w \delta \nabla \hat{v}(S,w)$
    $\theta \leftarrow \theta + \alpha^\theta I \delta \nabla \ln \pi(A|S, \theta)$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
Advanced Policy Gradient Methods

More advanced policy gradient methods:

- Natural Policy Gradient
- Trust Region Policy Optimisation
- Proximal Policy Optimisation
- Deterministic Policy Gradient

(Search on Google Scholar)
Reading

Required:

- RL book, Chapter 13 (13.1–13.5)

End of examinable material. For extra exam revision, see Tutorials 8 & 9.

Optional: